FairCom®
Update Guide

For

c-tree Plus® V6.11

and the

FairCom® Server V6.11.36

Published by

FairCom Corporation
2100 Forum Blvd., Suite C
Columbia, MO 65203

(573) 445-6833

www.faircom.com

First Printing, December 2001

V6.11(1024b)(1213)

Table of Contents

1. Introduction

The FairCom Team is pleased to deliver the latest release of our V6 technology. This new release includes a number of new features and enhancements, most of which have come directly from YOU, our developer-partners. As you continue to work with our technology, including this new release, we encourage you to keep sending us suggestions for new features and enhancements. As the developers on the front lines, you are our best source of information on what is needed to succeed.

The information in this update guide pertains to the following products:

- **c-tree Plus** File Handler V6.11
- **FairCom Server** V6.11.36
- **FairCom Server** V6.11 Software Development Kit (see documentation in Chapter 14 of the **c-tree Plus Programmer's Reference Guide**)

This Guide is made up of the following sections:

Chapter 1. Introduction

Chapter 2. New Features and Enhancements – Features and enhancements added in this release

Chapter 3. Fixes – Critical and minor issues addressed since the commercial release.

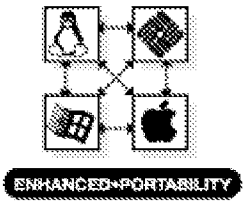
FairCom has been working hard and we are proud to bring these new features and enhancements to you. Thank you for your commitment to FairCom technology and we hope you find great value within this release. As always, if you should have any questions or special needs not addressed in this release, we encourage you to contact us at **info@faircom.com** or call your nearest FairCom office.

2. Features and Enhancements

Our V6.11 release includes a number of new features and enhancements to our technology. As you browse through the following pages, you'll discover that we've added support for new platforms, enhanced support for existing platforms, and added a number of diagnostic tools that will assist you with troubleshooting and support.

Because of the breadth of development projects that utilize our technology, some of these enhancements may not apply to the projects you are currently working on. Nonetheless, we encourage you to read the entire list carefully to stay abreast with the new developments in our technology.

2.1 New Platform Support



FairCom continues to lead the industry in the number of platforms that we commercially support! In fact, we take pride in the fact that we support everything from embedded operating systems such as QNX and LynxOS to desktop operating systems like Windows ME and Mac OS, to server operating systems like Solaris and HP-UX.

We are continuing that tradition in this new release with added support for QNX RTOS v6.

QNX RTP Platform Added

FairCom has ported c-tree Plus and the FairCom Server to the latest operating system from QNX Software Systems, QNX RTOS v6. The QNX realtime platform (RTP) is built on the QNX Neutrino realtime OS, one of the most popular and advanced real-time operating systems on the market.

FairCom has been a development partner with QNX Software Systems for many years and our solutions have been deployed together in many different industries including the industrial, medical-equipment, telephony, and automotive fields.

QNX currently offers its real-time operating system to developers at no charge from its web site at www.qnx.com.

2.2 Enhanced Platform Support

In addition to the two new platforms that we are now supporting with this release, we've enhanced our support for a number of additional platforms.

Updates for Mac Developers

2001 was a good year for Mac developers. The release of Max OS X marked a significant milestone for Apple, and revitalized the Mac developer community.

FairCom included support for Mac OS X in our V6.10 release, before Mac OS X was commercially released. So in case you missed the news in our last release and in case you missed our booth at Apple's World Wide Developers Conference, we do formally support Mac OS X.

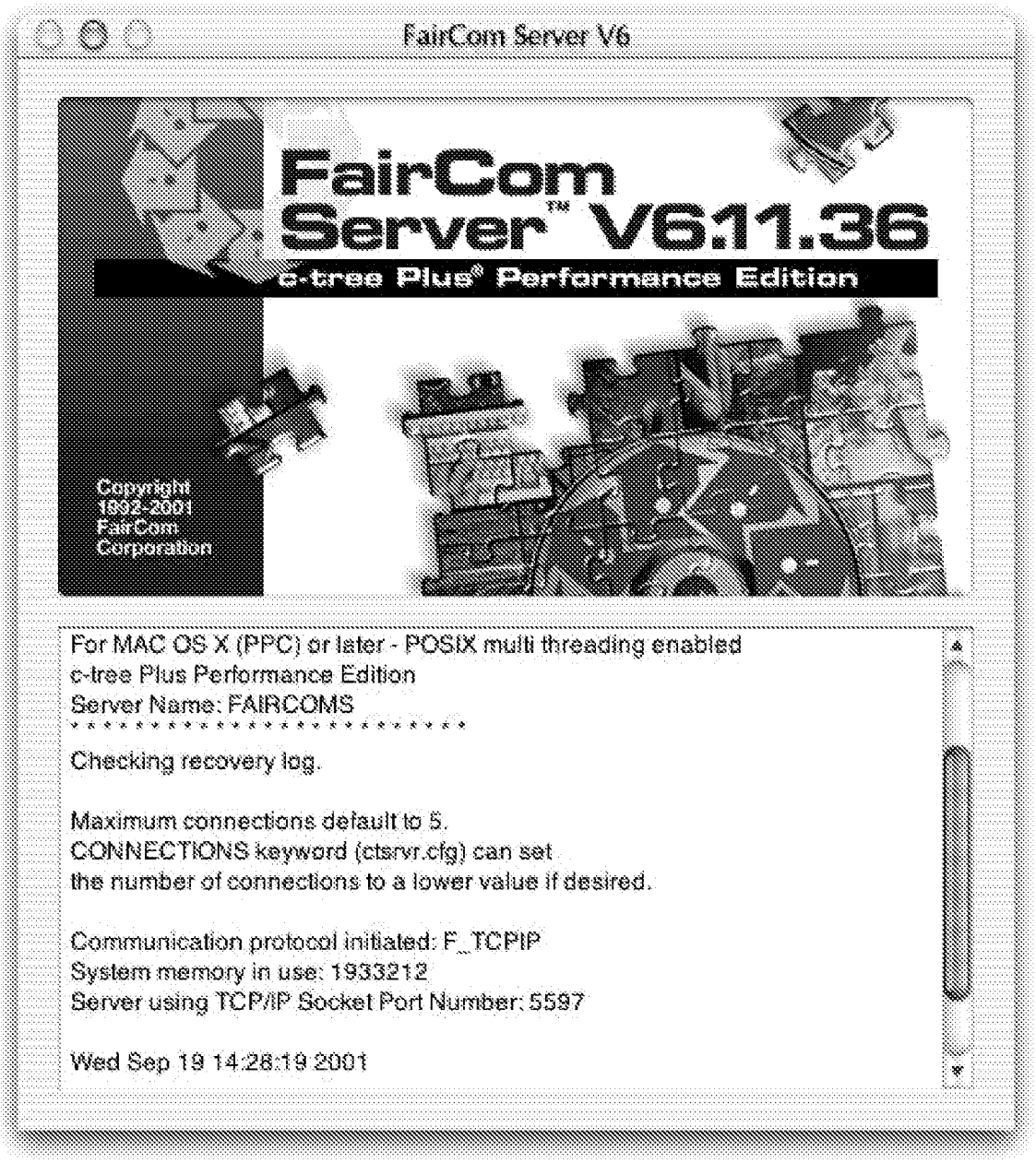
Carbon Support

With the release of Mac OS X, Apple introduced the Carbon API. Carbon is the set of programming interfaces derived from earlier Mac OS APIs that can run in Mac OS X. Some of these APIs have been modified or extended to take advantage of Mac OS X features such as preemptive multitasking and protected memory.

c-tree Plus Standalone models have been formally "carbonized" to support building applications for both the Mac OS 9 and Mac OS X platforms. Updated CodeWarrior projects include the proper settings to control this new support.

Mac OS X Cocoa Interface for FairCom Server

In anticipation of the commercial release of Apple's recent Mac OS X, we gave the **FairCom Server** a face-lift on this platform. A new graphical front-end interface, using Apple's Cocoa development tools, has been added to the Server on Mac OS X.



Shared Libraries for Mac OS X

The current make files address implementing shared libraries for Max OS X. This feature allows multiple applications to share a single **c-tree Plus** library, minimizing executable size.

Standalone Multi-threaded Model Enhanced

Support for Multi-Threaded multi-user applications (FPUTFGET) has been improved for the Apple Mac platform. A number of code cleanup and adjustments were involved in verifying this support and enhance performance and stability.

Server Shutdown Prompt Adjusted

The **FairCom Server** for Apple Mac shutdown screen, which indicates if any users are still logged onto the Server, now accepts keystrokes to select either CANCEL or YES. Prior to this change, the mouse was required. This allows a keystroke-input stream to automate shutting down the Server.

Symantec 7 Projects updated

At the request of a customer, the old Symantec 7 projects were resurrected in order to satisfy a legacy need. The code modifications were minimal which is a testament to the well-coded heart of **c-tree Plus**.

Updates for Windows Developers

FairCom Server for Windows continues to provide a powerful solution for developers deploying on Windows platforms, combining the power and flexibility of **c-tree Plus** in a low cost, low maintenance database server.

If you are still deploying using our standalone models, we encourage you to try our development servers that are included on the **c-tree Plus** V6.11 CD. We know you and your customers will appreciate the added features that our server solutions can bring to your application.

Dynamic Message Monitor and Function Monitor

Menu options have been added to the **FairCom Server** for Windows in order to dynamically turn on/off both the function monitor as well as the console Message windows. This adjustment will make it easy for developers to activate the function monitor without having to stop and then restart the Server.

Server Shutdown as Windows NT/2000 Service

The **FairCom Server** can be run as a service on Windows NT/2000. If the properties are configured to allow interaction and the user clicks the Control-Shutdown icon, the Server now prompts the user to Continue or Cancel before stopping. If the user selects Continue, the service shuts down properly and the Server comes down cleanly without further interaction. In the past, the Server and service were simply shut down without a prompt.

Additional Platform-Specific Updates

We've made a few additional platform-specific changes that may affect the platforms you are utilizing.

NetBSD, FreeBSD, Tru64, and Solaris (Intel) allow 4 GB Files

The default for the NetBSD, FreeBSD, Tru64, and Solaris (Intel) platforms has been updated to support up to 4 GB files. Support for files larger than 4 GB is offered with **c-tree Plus** Professional V7.

HP-UX 64 bit Compile Option Added

The HP9000 options have been updated in the m-tree make system in order to support 64 bit compiling on the HP-UX 11 operating system. Now developers may choose between 32 bit or 64 bit compile options. Shared libraries are supported in both cases.

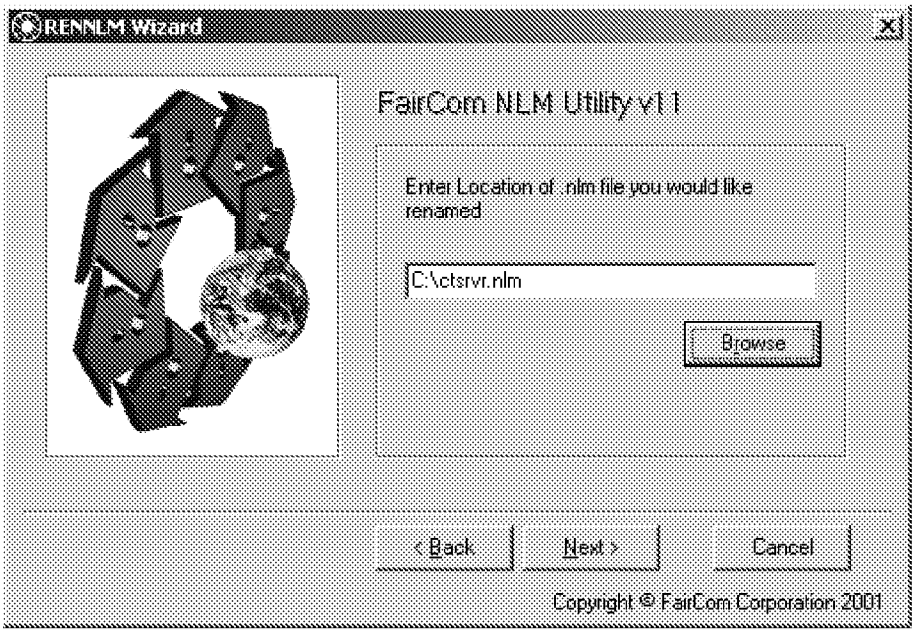
Shared Library Support for HP-UX 11 and SCO OpenServer 5

The m-tree system now includes shared library options for HP-UX 11 and SCO OpenServer 5. This feature allows multiple applications to share a single **c-tree Plus** library, minimizing executable size.

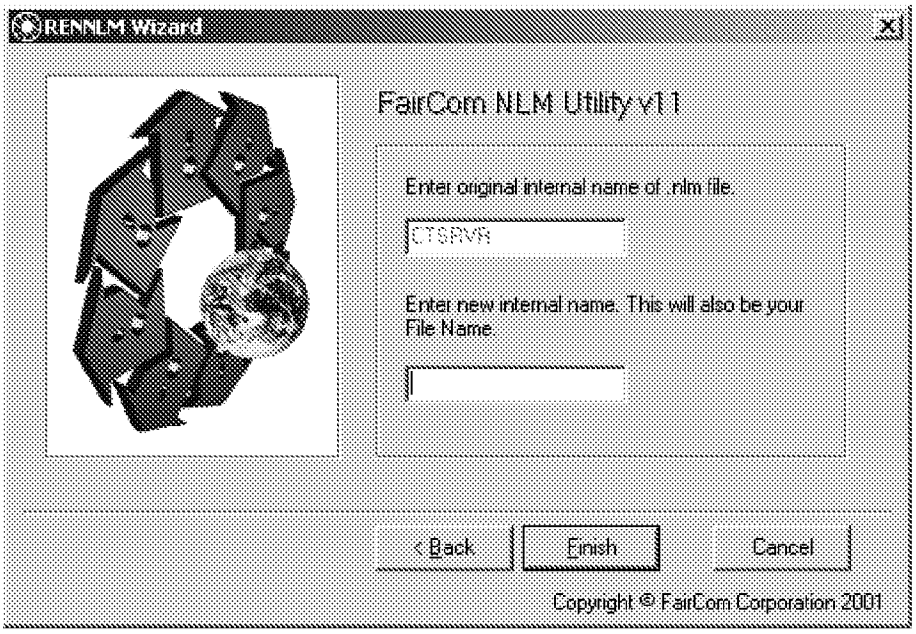
New GUI NLM Rename Utility

Developers who wish to run more than one instance of the **FairCom Server** for Novell are already familiar with the *rennlm* console utility. Now the new GUI version of this utility is available, as shown below.

Simply provide the original file name from which the utility can generate a new NLM (by browsing the file folders, if desired):



Then provide a new name and click Finish:



The new NLM is placed in the same directory as the original. See "Running Multiple FairCom Servers on One Machine" in the **FairCom Server Administrator's Guide**.

2.3 Key Compression for Key Lengths Over 255

Prior to this change, key compression was supported for key lengths up to `MAXLEN` (255). Users who applied key compression to key lengths greater than 255 could have experienced unpredictable results.

The compression logic now properly handles `MAXLEN` values greater than 255.

NOTE: Although it is possible to compress keys larger than 255, only the first 255 and/or the trailing 255 characters are compressed. The internal logic uses a one-byte length value to indicate the number of characters compressed.

NOTE: While `MAXLEN` can be changed for Standalone models, the **FairCom Server** defaults to a maximum key length of 255 bytes. However, the **c-tree Server** from **c-tree Plus Professional** V7 supports a `MAXLEN` of 1024 bytes. If you need keys larger than 255 bytes in a client/server environment, consider upgrading.

2.4 FairCom Server Detects Dropped Clients

The **FairCom Server** normally recognizes when a client disconnects. However, the Server relies on a chain of events controlled by the operating system in order to recognize the disconnection. The client computer must notify the Server host computer that the connection has been dropped. For example: When a user closes an application, the socket is closed by the operating system, which sends a message to the Server host machine. However, if the network connection is temporarily interrupted or if the client machine is powered down suddenly, this message is not sent and the Server host machine can't recognize that the client connection has dropped.

With the `COMPATIBILITY TCPIP_CHECK_DEAD_CLIENTS` keyword in the Server configuration, the **FairCom Server** detects when a TCP/IP client has dropped. Every 120 seconds, the client connection socket is rechecked to ensure that it is still a valid communications channel. If this connection is found to be invalid, the Server will terminate this connection.

This functionality is not supported on the FairCom Servers for Mac, OS/2, or early Linux (kernal less than version 2.036) at this time.

2.5 Logon Timeout Enhanced

The Server `LOGON_FAIL_TIME` keyword now supports a value of –1, which indicates that there should be a permanent "log-on block" placed on a user. To help put this in context, the `LOGON_FAIL_LIMIT` keyword specifies the number of "strike-outs" or "failed logon attempts" that the Server allows before "blocking" additional logons for a specific user. The `LOGON_FAIL_TIME` keyword specifies the time that a user should be prevented from logging on after the User ID has been invalidated. This change allows this invalidation period to be forever/permanent or at least until the Server ADMIN intervenes.

2.6 Server Detects Inactive Clients

A new keyword SESSION_TIMEOUT has been added that instructs the Server to remove TCP/IP connections after a specified number of seconds have elapsed without any activity. The server checks connections periodically (every 5 minutes on most platforms) to determine whether the SESSION_TIMEOUT interval has elapsed and, if so, whether the connection should be terminated for inactivity. This allows clients with no activity to be disconnected from the Server automatically.

```
SESSION_TIMEOUT <seconds>
```

The theoretical maximum number of seconds is over 2 billion, so any practical values are supported. For example, a value of 1800 would remove the connection after 30 minutes. This new keyword is off by default.

This feature has been tested on Windows, Novell, Linux, Mac OS 9, and Mac OS X.

2.7 Diagnostics Enhancements

The following features and enhancements provide additional options for developers troubleshooting their applications.

TRAP_COMM Utility Added

The DIAGNOSTICS TRAP_COMM Server keyword allows developers to 'record' the communication traffic coming in to a Server and play it back with the *cttrap* utility.

cttrap – Communications Trap Playback utility

cttrap <TrapCommLogFileName> [<ServerName>]

cttrap is a multithreaded client application that 'plays back' a TRAP_COMM log file. Whenever a Multithreaded Client library is created, cttrap will also be generated. The default TRAP_COMM file name is *TRAPCOMM.FCS*.

DIAGNOSTICS TRAP_COMM

When activated, the DIAGNOSTICS TRAP_COMM keyword instructs the **FairCom Server** to log incoming communications packets to *TRAPCOMM.FCS* prior to execution. This log can be played back using the cttrap utility and a debug Server to observe the results of the client requests, allowing the developer to exactly duplicate and repeat client activities.

The trap file, *TRAPCOMM.FCS*, is created in the Server directory by default. To prepend a path onto the trap file name (say to route it to a separate disk), add an entry of the form DIAGNOSTIC_STR <trap file path>. For example, if DIAGNOSTIC_STR /bigdisk/ were in the configuration file, then the trap file would be */bigdisk/TRAPCOMM.FCS*.

DIAGNOSTICS LOWL_FILE_IO Keyword

The `DIAGNOSTICS LOWL_FILE_IO` Server keyword logs low-level system errors into the Server status file, *CTSTATUS.FCS*. Although client applications have access to system errors through *sysiocod*, it is useful to see these errors logged on the Server side. For example: An end-user has problems opening a file. The end-user copied the data file from a CD-ROM onto the hard disk leaving the file marked "read-only". When the user attempted to open these files, open errors were generated within the application. Adding the `DIAGNOSTICS LOWL_FILE_IO` keyword directed the Server to log the "not authorized" operating system error to *CTSTATUS.FCS*. This pointed the user to the "read-only" issue.

Debug Node I/O Diagnostics Enhanced

A number of internal improvements provide more detailed diagnostics information related to Index Node I/O. Periodic reports of a rare un-reproducible `BNOD_ERR(69)` error justified this effort where we applied internal diagnostics designed to report internal states when this error occurs.

Adding the `#define DBGnodeIO 10000`, where 10000 is the size of an internal buffer, before building the **FairCom Server** causes the Server to maintain a circular buffer of information that will offer FairCom insights into this problem. If the 69 error is encountered, the Server writes the buffer to the *CTSTATUS.FCS* file.

2.8 Other Enhancements

Function Monitor Enhancements

The FUNCTION_MONITOR window now displays the file name in addition to the User ID and function name to allow Server Administrators to monitor activity for specific files.

The optional FUNCTION_MONITOR log (created when a file name is a parameter to the FUNCTION_MONITOR keyword in ctsrvr.cfg) now includes the **c-tree Plus** return value (error number) for every **c-tree Plus** function called by the client. This option gives the ability to execute a client application, as is, and inspect in this log the return values of all **c-tree Plus** function calls made by the client application.

GetServerInfo Function Exported

At the request of a customer, the function **GetServerInfo** has been added to the exported functions defined in a DLL. This function might be used in a client application when implementing the Server Broadcast feature. Without this export, the function would come up undefined.

Superfile Compact Utility Adjusts Sector Size

The superfile compact utility has been enhanced to allow the user to provide a sector size for the superfile. Now ctscmp can be used not only to compact superfiles but also to change their sector size. This change allows Administrators who created a *FAIRCOM.FCS* using one PAGE_SIZE setting to restart the Server with a new PAGE_SIZE setting but use the current *FAIRCOM.FCS*. The command-line usage is now: `ctscmp [Y] [new_sect_size]`, where the optional argument *Y* compacts the file without prompting to confirm, and the optional argument *new_sect_size* is the sector size of the resulting file.

DoBatch BAT_KEYS Option Skips Missing Key Values

The behavior of the **DoBatch** call was changed when using the BAT_KEYS option. **DoBatch** now returns records found for each and every key given. A customer reported unexpected behavior using the previous version of **DoBatch** when passing in a set of key values for input, the BAT_KEYS option. If one of the keys in the set did not exist, the batch terminated without retrieving any other values. This is not a common problem since the set of keys passed in is frequently based on keys known to exist (e.g., from a previous **DoBatch** request).

New IFIL-based Rebuild Utility – ctrbldif

A new rebuild utility using the IFIL definitions stored in the header of a file has been added to **c-tree Plus**:

```
ctrbldif DataFileName [<UserId>] [<UserPassword>]
                      [<ServerName>] [-purge] [-<sectors>]
```

ctrbldif reads the IFIL structure from *DataFileName* and calls **RebuildIFileXtd** to rebuild *DataFileName* and its associated indices.

UserID, UserPassword, and ServerName are only needed for client versions of this utility. FairCom recommends building this utility as a Single-user Standalone application.

-purge indicates that duplicate records should be purged.

-<sectors> is the sector size to use. The sector parameter is especially useful if you need to adjust a file's `PAGE_SIZE` to match the **FairCom Server**.

New IFIL-based Compact Utility – ctcmpcif

```
ctcmpcif DataFileName [<UserId>] [<UserPassword>]
                      [<ServerName>] [-purge] [-<sectors>]
```

ctcmpcif reads the IFIL structure from DataFileName and calls **CompactIFileXtd** and **RebuildIFileXtd** to compact and rebuild DataFileName and its associated indices. If ctcmpcif cannot extract the IFIL from the target file, it will ask for the name of another copy of the file from which to extract the IFIL information.

UserID, UserPassword, and ServerName are only needed for client versions of this utility. FairCom recommends building this utility as a Single-user Standalone application.

-purge indicates that duplicate records should be purged.

-<sectors> is the sector size to use. The sector parameter is especially useful if you need to adjust a file's `PAGE_SIZE` to match the **FairCom Server**.

ctMAXSORTBUF placed in ctoptn.h now overrides CTSORTBUF

A new `#define ctMAXSORTBUF` has been added in order to allow the user to override the default buffer size used by the **c-tree Plus** rebuild. `#define ctMAXSORTBUF` determined the amount of memory used to buffer key sorts when rebuilding the indices associated with a data file. A larger sort buffer, in most cases, will result in a quicker rebuild.

By default, this value is 16000. On many occasions, FairCom's technical support team has found it necessary to advise the customer to increase this value. Handled as a UINT, `ctMAXSORTBUF` has a theoretical maximum of 4GB or 4294967295 bytes, but the optimum value will not exceed the available system memory.

Example:

```
#define ctMAXSORTBUF 64000
```

3. Fixes

We recognize how important it is to know what has changed between releases so that you can be aware how changes will impact your particular project. Accordingly, we've listed every significant bug that has been resolved since the last major release. (Who else but FairCom would give you this level of detail!)

As you'll see, we've separated out these issues into items that we feel are critical and items that we feel are relatively minor. Each entry is marked with the build date that included the fix, and entries are listed in chronological order.

3.1 Critical Fixes

Critical issues represent serious problems resulting in a crash or data corruption. Though most are unlikely to occur, they have serious consequences when they do.

Windows 95 Standalone Multi-user Crash Resolved (1204)

A problem that only affects Windows 95 applications using Standalone Multi-user libraries has been resolved. c-tree Plus includes dynamic detection of the Windows redirector version (vredir.vxd), in order to determine whether or not c-tree Plus must use I/O locking to prevent the redirector from caching data for network file I/O operations. The function that scans the vredir.vxd file for version information, which is only called when running on Windows 95 systems, may attempt to read past the end of a memory buffer, causing an access violation. This code has been adjusted to ensure this not longer happens.

SwitchCtree Core Dump Fixed (0410)

FairCom corrected a potential for addressing a NULL pointer within **SwitchCtree**, which could cause an application to invoke an exception error.

Dynamic Dump Roll Forward Error Resolved (0424)

An issue has been corrected in the dynamic dump restore utility, ctrdmp, that could cause a `RFCK_ERR(510)` while attempting to roll forward from a dynamic dump restore. The issue involved the internal checkpoint handling. There was a chance that a roll forward following a dynamic dump restore could start from the wrong checkpoint. This was not an issue with a roll forward from a standard backup. This is considered a critical problem. All users who utilize the ctrdmp and ctfdmp utilities to roll forward from a dynamic dump restore should update immediately.

HP-UX 11 Update (0424)

A problem related to incorrect compile switches on HP-UX 11 has been resolved. Customers who are developing applications for the HP-UX 11 platform should update immediately.

Windows Shared Memory Logon Issues Resolved (0509)

A subtle problem related to logging on a **FairCom Server** has been resolved. When rapid logon/logoff sequences were pursuing the **FairCom Server**, the internal shared memory pipe response would return a "busy" error, thus denying the clients the ability to logon. By adding internal retry logic, this issue is now resolved. Any users who experienced occasional logon failures during high volume of logon/logoff activity should appreciate this fix.

BAT_RET_KEY and uTFRMKEY (0509)

DoBatch called with `BAT_RET_KEY` was returning keys in the byte-order native to the client. This has been adjusted to work as documented, returning keys in native index format (i.e., `HIGH_LOW`).

To restore the old behavior (keys returned in client numeric format) in Standalone models, add `#define ctBEHAV_BATUTFRM` at compile time.

To restore the old behavior (keys returned in client numeric format) with the Server, add `COMPATIBILITY BATCH_UTFRMKEY` to the Server configuration.

Incorrect End-of-File Value Restored When 4GB Limit Exceeded (0525)

When an attempt was made to exceed the 4 GB file limit for a non-huge file, the file extension routine would correctly detect that the limit was exceeded and return a `FULL_ERR(39)`. Similarly, on a disk full check, the `FULL_ERR(39)` would be returned if the volume was full. However, the logical end-of-file value in the header was not being restored properly. After the error occurred, new records could be added to the file, but they would overwrite the beginning of the file. Note that the information that was overwritten would not necessarily start at the absolute beginning of the file, depending on how the new space request wrapped around the 4 GB limit. This issue has been resolved so that the header value is updated correctly.

Corrected PREIMAGE_DUMP catend (0717)

A dynamic dump restore, *ctrdmp*, could terminate with an L64 error on `PREIMG` files dumped with the `PREIMAGE_DUMP` keyword. A status value was altered to correct this issue.

Threading Semaphore (0717/0816)

An internal semaphore issue, which caused the Server to fail under QNX RTP, has been resolved. This change was applied to all Unix platforms in the 0816 release.

Automatic Recovery Index Errors (0808)

An issue that could result in index errors, such as `BNOD_ERR`(69), during or after automatic recovery or dynamic dump restore has been fixed. Automatic recovery of indices with multiple members infrequently resulted in damage to the headers. Data files were not affected and a rebuild corrected the issue.

Dynamic Dump Restore BNOD_ERR(69) – (1018)

An issue with the c-tree Server restore process caused selected indices to not be addressed properly during the restore procedure. The pivot point of the problem was a subtle issue related to when transactions completed and when the dynamic dump process issued its final checkpoint. Specifically, an index that had just experienced activity under transaction control and had been committed was being detected as "in-order", when, in fact, it required additional consideration by the ctrdmp recovery routine. Data files were not affected and a rebuild corrected the issue.

3.2 Other Fixes

The following other issues were corrected.

NLM Invalid LOCAL_DIRECTORY Fix (1029)

An issue related to an invalid or non-existent directory name being used with the `LOCAL_DIRECTORY` server keyword has been resolved. The **FairCom Server** assumes that when a directory name is specified for the `LOCAL_DIRECTORY` keyword, that that directory already exists and has proper permissions set in order for the Server to access this directory. Prior to this fix, the Server did not contain validation logic that verified the proper existence of this directory name. If an invalid directory name was specified, the Server would improperly terminate causing the Netware server to hang. A check has been added in order for the Server to validate this directory and properly terminate if invalid.

Shutdown Adjusted for Long Client Cleanup (1204)

Under qualified circumstances it was possible for the Server to improperly terminate during shutdown. If a client thread required an exceptionally long time to exit (for example, in the middle of a very long transaction) after indications that the Server was shutting down, it was possible for the Server to free resources used by the client before the client completed processing. This caused an exception violation. This adjustment ensures that these excessively involved clients are given adequate time to finish before the Server terminates.

Rebuild IEOF_ERR(519) Fixed (1204)

A situation where the rebuild of an index containing resources returned IEOF_ERR(519) was corrected.

Unexplained Delete and Unlock Errors Corrected (1231)

A qualified un-common scenario where record updates, deletes or unlocks resulted in an unexplained error has been addressed. Because of the resulting errors, this issue would be obvious if it affected an application. This issue was qualified to the use of fixed length records under transaction control when using any variation of the `ctKEEP` functionality (i.e., `ctKEEP`; `ctKEEP_OUT`; `ctKEEP_OUT_ALL`) during transaction **Commit** or **Abort**. Users who use the `FREE` locking mode for **Commit**

or **Abort** are not affected by this problem. Record updates and deletes would most often experience a KDEL_ERR(4) error, while record un-lock calls would receive a UDLK_ERR(41).

Users who take advantage of the **c-tree Plus** automatic ISAM transaction support (i.e., **SetOperationState**(OPS_AUTOISAM_TRN)) should note that the transaction commits use ctKEEP or ctKEEP_OUT.

For fixed length data records, deleting the record under transaction control can lead to the actual lock on the deleted record being released but the user's private lock table still considers the lock as owned by the user. This will occur if the commit attempts to ctKEEP or ctKEEP_OUT_ALL, or if no **LockISAM** state is turned on at delete time (say because OPS_LOCKON_GET was used to get a record lock) and a commit with ctKEEP_OUT is called. Then if this same user attempts to get a lock on this same record subsequently, the user lock table entry keeps the server from acquiring a system lock (because the system trusts the user lock table). For variable length records, a commit with ctKEEP or ctKEEP_OUT_ALL can lead to the same situation. This issue has now been resolved.

KEEP_OUT_ALL Behavior with RestoreSavePoint (1231)

If a lock is obtained on a fixed length record before a transaction, then after **Begin** and a **SetSavePoint** the record is deleted, and then the delete is undone by a call to **RestoreSavePoint**, the lock will remain. Prior to this fix, this lock had been freed. This change preserves the lock, as one would expect.

DLOK_ERR on Record Add Corrected (1231)

A subtle problem which could result in an **AddRecord**, **AddVRecord**, or **NewData** receiving a lock error (DLOK_ERR(42)) has been addressed. This situation could happen if one user issues a "put-to-sleep" (blocking) type of lock, and while waiting for the lock, the associated record is deleted by another user. Once the other user frees his lock, then the deleted record position (now placed in the deleted record stack available for re-use) is locked by the user who was waiting for a lock. Then, if a different user issues any form of add record, which calls NEWREC() to obtain a new record position from the delete stack, this add record will fail with a DLOK_ERR(42) because this new record position is locked.

Although this situation seems common, it is actually highly unlikely, and will typically only arise under extreme conditions where the same record is constantly being locked (in blocking mode), deleted, and reused over and over.

Communications Code Cleanup for Tru64 (0126)

Clients compiled on a Dec-Alpha/Tru64 could not connect to the Server. Adjustments were made to **c-tree Plus** communication code to solve this concern.

Dynamic Dump Restore KLNK_ERR Resolved (0201)

An internal issue related to Dynamic Dump restore failing with a `KLNK_ERR(25)` has been resolved. This problem applies to all platforms.

WatCom Compiler Correction (0201)

A number of subtle issues related to the WatCom compiler have been cleaned up. Any users who experienced syntax errors when using Watcom should note these adjustments.

Server Shutdown with Many Clients Fix (0216)

When a large number of clients (over 100) are attached to the **FairCom Server** and the Server is shut down, there were isolated occurrences where the Server would not give all clients adequate time to exit. This would cause the Server to core dump. Additional changes have been applied to the shutdown timing of the primary thread to avoid this situation.

Windows Server Thread Exit Memory Leak (0216)

A small memory leak has been fixed that affected both the **FairCom Server** and the Bound Server model. A potential leak of 116 byes was reported for each terminated thread. Therefore, Servers that are continually servicing users logging on/off over a long period of time (without any Shutdown) should take note of this issue.

FairCom Server Threading Issue on AIX 4.3 Resolved (0223)

An issue in the Server internal defer logic on AIX 4.3 has been corrected. Users experiencing problems with Dynamic Dumps or Server shut down should consider upgrading.

Obscure RestoreSavePoint Error (0314)

An obscure issue which might have caused an unexpected I/O error such as `RRED_ERR(407)` after a **RestoreSavePoint** operation has been resolved. This is a moderately serious bug, but one not likely to occur. The following pseudocode sequence would lead to the problem:

```
Begin
update record or resource
SetSavePoint
SetSavePoint
update same record or resource
ClearSavePoint
RestoreSavePoint
```

After this sequence the first update will be gone as well as the second update, which was purposely undone by the **RestoreSavePoint**. In practice, virtually no one will perform this sequence explicitly. However, internally, when updating FairCom resources, we use **SetSavePoint/ClearSavePoint** to isolate our behind-the-scenes

updates from the user's explicit calls. This situation is most likely to occur, if at all, during a file creation sequence involving transaction dependent files in which the **Begin** is called before the file create call, the user calls **SetSavePoint** after the create, the user then calls something like **PutDODA** and then **RestoreSavePoint**. This leaves the resource header all FF's which causes the RRED_ERR(407). This problem has been rectified.

Win32 FPUTFGET Read Error Corrected (0314)

The default setting in **c-tree Plus** is to disable the I/O LOCKING that was previously enabled for all Win32 environments in the prior release, controlled by the macro ctPortWFW. **c-tree Plus** automatically enables the I/O LOCKING if the platform does not have the proper "redirector" (vredir) installed. Generally this code will only be enabled for Windows 95 or possibly a version of Windows 98 that doesn't have a current vredir installed.

In such a mixed environment (FPUTFGET) where one or more machines are doing I/O LOCKING and the others are not, there is a potential for a READ_ERR(36) error because the locks applied by the I/O LOCKING are to the actual data area of the file. Under DOS/WINDOWS, areas of a file locked are not even readable by another process, which produces the READ_ERR(36). In this situation the READ_ERR(36) can be viewed the same as a ITIM_ERR(160) error and the application needs to retry or delay and retry to perform the **c-tree Plus** function. If all clients are using the I/O LOCKING, then the retry is automatically handled inside the I/O LOCKING code.

In order to solve this problem, we applied a fix so that if a read or write fails and the system error code is ERROR_LOCK_VIOLATION (indicating an application that is not using I/O locking was locked out by an application that is using I/O locking), the non-I/O locking application attempts to get a lock and then retries the read or write operation if the lock was successful. After applying this fix, the READ_ERR(36) no longer occurred.

Windows 98 Shutdown Issue Resolved (0314)

Prior to this fix, under Windows 98, if a **FairCom Server** was running in Tool-Tray mode and the user selected "Start Menu/Shutdown", the **FairCom Server** terminated but the Windows operating system did not finish its shutdown procedure. Internal adjustments were made to allow the **FairCom Server** to indicate to the operating system that it was okay to continue the shutdown process.

Dynamic Dump Recover with LOG_ODD/LOG_EVEN (0410)

Prior to this fix, dynamic dump recovery (*ctrdmp*) mishandled the LOG_EVEN and LOG_ODD keywords, returning LOPN_ERR(96).

Mac Multi-Threaded Standalone (0410)

A number of subtle issues related to implementing the Multi-threaded Standalone model on the Apple Macintosh have been cleaned up. Users experiencing any issues on the Mac with this model should update to this newer set of code.

Multi-Threaded Standalone Issues Resolved (0420)

Multi-Threaded Standalone issues on 64 bit platforms related to **RegisterCtree**, return type casting, and mutex structure sizes have been fixed. Users experiencing any type of instability in this model on 64 bit platforms should consider updating.

Eliminate terr's Corresponding To Corrupt Indices (0424/0717)

A number of internal "terr" errors (terminating errors) have been adjusted within the **FairCom Server**. Prior to these adjustments, if a specific user (thread) encountered an internal integrity problem, the Server would shut itself down in what FairCom calls a "terr" situation. The terr errors addressed were 218, 220, 240, 7214, and 8214, which correspond to corrupt indices. If a user thread encounters a bad index condition, the Server sends the application an error rather than terminating. These types of errors a rare and do not affect most users.

Dynamic Dump Restore FTYP_ERR(53) Corrected (0424)

An improper `FTYP_ERR(53)` error generated when a file larger than 2GB was restored with the ctrdmp utility was corrected.

Variable-Length Rewrite Serial Number Behavior Changed (0424)

c-tree Plus now rewrites variable-length records containing `SRLSEG` segments without incrementing the serial number in the file header. Prior to this change a variable-length record rewrite that moved the record in the file incremented the internal serial number in the header of the file. The serial number in the record was not changed, but this caused a gap in the sequence of serial numbers being generated.

DoBatch with Heterogeneous Clients (0525)

An issue was corrected with **DoBatch** when using a key containing a numeric segment in a heterogeneous environment. **TransformKey** was not called, which sets numeric key segments to be in the correct byte order by the time the key gets to the Server. **Note:** This may not have been a broad problem since most applications use the batch routines (or the set routines) with keys containing non-numeric segments.

Obscure Bad Node Split Issue Repaired (0525)

A unique scenario was discovered that could result in an improperly formed index that might produce a subsequent `BIDX_ERR(527)`. This problem only occurs when:

1) A transaction controlled index holds unique keys;
2) The nodes may only have room for less than 8 key values;
3) An attempt is made to delete and re-add the same key at a different record position within the same transaction (e.g., rewriting a variable length record with an unchanged unique key, and the new record image does not fit in its original file position);

4) Either DECADD is explicitly chosen in a call to **AddKey** (not very likely), or in V7 the adaptive node split logic requests a DECADD node split. (DECADD implies keys are being added in decreasing key-order.);
5) The deleted key value (still pending commit) and the re-added key value (still pending commit) occupy the first two positions in the leaf node.

If ALL of these conditions are met, the node split logic attempted to split the node before the first key value, which is a meaningless position within the node. This is a V6 and V7 issue although it is much more **unlikely** to occur in V6.

Client Hang Superfile Member RebuildIFile (0525)

An issue with a client application appearing to hang when calling **RebuildIFile** for a superfile member has been corrected. An **OpenCtFileXtd** call (performed to verify the superfile member exists) did not "or" in the ADMOPEN file mode bit. Without this bit, the **OpenCtFileXtd** call caused the Server to believe additional data will be returned by the **RebuildIFile** call. This causes the client and server to get out of synch in some non-Windows communication environments.

UNIFRMAT with Conditional Index (0525)

An internal issue has been resolved related to the use of Conditional Indexes with UNIFRMAT. Any user who might have used this unique configuration and bumped into any problems are encouraged to update to this latest set of code.

Server Shutdown if first COMM_PROTOCOL fails (0525)

An issue was resolved related to shutting down the server if the first COMM_PROTOCOL defined in the Server configuration file (*ctsrvr.cfg*) fails to load. When a communication protocol fails to load at startup, the Server issues a warming but continues using any other defined protocol. In the case of this error, when the request was made to shut down the server, an error would be generated preventing a clean shutdown.

Additional DEBUG Information for 8521/8522 (0717)

In the event of a terr(8521) or terr(8522), additional information is output.

CTFLUSH Read-only Files Issue Resolved (0717)

A customer reported a problem while running from a CD. Because the files were "read-only" the customer was experiencing an FSAV_ERR(49) error when the CTFLUSH function was being called. This issue had been fixed.

Dynamic c-tree Plus DLL Switching Issues Resolved (0808)

A few issues related to the use of dynamic **c-tree Plus** DLLs with different Operational Models have been corrected. Users who experienced problems while trying to support simultaneous **c-tree Plus** models (e.g., Client and Standalone) in separate DLLs from the same application should benefit from these fixes.

Visual Basic 16 bit (0808)

A number of small syntax issues have been resolved related to 16 bit use with Visual Basic. Although this might be consider legacy, FairCom is happy to continue to support the models used by all our customers.

TCP/IP – 64Bit 'localhost' Issue (0816)

TCP/IP clients in 64-bit environments could only connect to FairCom Servers from the host machine, or localhost. An internal correction changed the client from treating the IP address as a 32-bit value, to a 64-bit value when appropriate, allowing a proper connection.

TCP/IP – Better Support for Many Simultaneous Logons (0816)

The FairCom Server now handles a larger number of simultaneous logon attempts without error.

CLIFIL and DELIFIL unresolved in Server SDK (0908)

The functions **CloseIFile** and **DeleteIFile** are no longer unresolved when using the **FairCom Server SDK**.

Borland v4 (BC4) and Pharlap DOS286 (0908)

At the request of a customer, FairCom made some minor corrections to provide support for Borland C v4 with the Pharlap DOS286 extender.

OPNRFIL with Deferred Close FUSE_ERR (0914)

FairCom corrected an issue that caused **OpenFileWithResource** to return `FUSE_ERR` when opening and closing a large number of files with deferred file close enabled: **SetOperationState**(`OPS_DEFER_CLOSE,OPS_STATE_ON`).

HP-UX 11 File Descriptor Limit (0914)

FairCom added logic to allow the **c-tree Server** for HP-UX 11 to increase its file descriptor limit up to the operating system's hard file limit to avoid artificially limiting the number of files and connections available to the Server.

Cancel of Scheduled Dynamic Dump Crash Fixed (0918)

FairCom corrected a error that resulted in a Server crash when a scheduled dynamic dump was canceled.

Mac OS X 10.1 Compatibility (1108)

Some adjustments to the Cocoa interface and some shutdown details were required to allow the FairCom Server for Mac OS X to shut down properly under V10.1.

End Of Notes